(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 984 390 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
08.03.2000 Bulletin 2000/10

(51) Int Cl.[7]: **G06K 19/06**

(21) Application number: **99306875.8**

(22) Date of filing: **27.08.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **31.08.1998 US 144251**

(71) Applicant: **XEROX CORPORATION**
Rochester, New York 14644 (US)

(72) Inventors:
- **Hecht, David L**
  Palo Alto, California 94303 (US)
- **Jared, David A.**
  Sunnyvale, California 94086 (US)
- **Flores, L. Noah**
  Soquel, California 95073 (US)
- **Stearnes, Richard G.**
  Felton, California 95018 (US)
- **Chang, Kenneth H.P.**
  Foster City, California 94404 (US)

(74) Representative: **Skone James, Robert Edmund**
GILL JENNINGS & EVERY
Broadgate House
7 Eldon Street
London EC2M 7LH (GB)

(54) Apparatus for providing location information in a multidimensional address space

(57) A user interface is implemented using visual indicia and a background for the visual indicia that encodes address information. The background appears visually as a stipple pattern, but is implemented using glyphs which form and address carpet that encodes address information uniquely identifying each area of the user interface. An image capture device is used to select an area of the address carpet that is under or near visual indicia of interest to the user. The image capture device captures the area of interest, and transmits the image area to a computer for processing. The computer first determines the proper orientation of the image, and then decodes the information encoded by the glyphs. The decoding results in an X, Y address identifying the location of the selected area in the address carpet. Based on the address, the computer may perform an operation associated with the area.

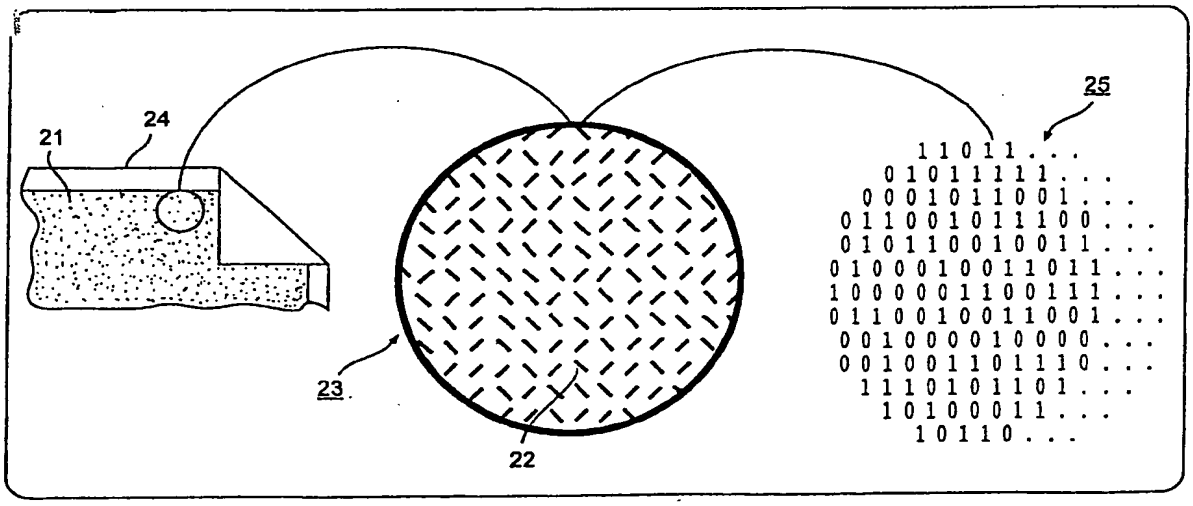

FIG. 1

EP 0 984 390 A2

Printed by Jouve, 75001 PARIS (FR)

Description

[0001] This invention relates to the construction of multi-dimensional address spaces and, more particularly, to self-clocking and address carpet glyph instantiations of such address spaces and disambiguation techniques for address carpets.

[0002] Glyphs are often encoded by respective elongated slash-like glyphs written on a two- dimensional, spatially periodic pattern of centers in accordance with a predetermined spatial formatting rule, with the individual glyphs being tilted to the left and right of vertical by approximately +45° and -45° for encoding logical "0's" and "1's", respectively. The mutual orthogonality of the glyph encodings for the two logical states of these single bit digital quanta enhances the discriminability of the code sufficiently to enable the embedded information to be recovered, even when the code pattern is written on a sufficiently fine grain pattern of center to cause the code pattern to have a generally uniform grayscale appearance.

[0003] Prior proposals have dealt with the general subject of constructing address spaces in the image domain. For example, address spaces may be constructed by encoding cyclical pseudo-noise digital bit sequences (sometimes referred to as "PN sequences") and other types of maximal length-like digital bit sequences (i.e., sequences of length, L, in which every N-bit long subsequence is unique) in two dimensional spatially periodic self-clocking glyph code patterns. One of the unifying themes of these prior proposals is that they recommend constructing two dimensional address spaces by mapping at least two bit sequences of the foregoing type into such a code pattern so that these bit sequences are encoded to propagate in predetermined directions along respective non-parallel lines of glyphs.

[0004] While it has been shown that the bit sequences need not be mapped into the code pattern in alignment with the principal axes thereof, it often is desirable to employ such mappings to reduce the computations that are required to determine the relative addresses of the glyphs in standard Cartesian coordinates (i.e., "x" and "y" parameters expressed in units of glyphs). Furthermore, it is known that the encodings of these bit sequences may completely or partially span the glyph code pattern, so it is to be understood that the addressing that is afforded by them is effective only within that portion of the glyph code pattern that projects onto both of the encodings. Moreover, the aforementioned bit sequences may be mapped into the glyphs on these non-parallel lines at unitary or fractional duty ratios, but it has been recognized that mappings preferably are spatially cyclical for applications in which it is desired to compute relative addresses for glyphs within the address space.

[0005] Every bit in a maximal bit length sequence resides at a predetermined uniquely determinable logical position within the sequence. Accordingly, an ordered index of integers is commonly employed to distinguish those bits from each other on the basis of their respective ordinal logical positions within the sequence. These positionally dependent index numbers can, of course, be employed for selectively addressing the glyphs which encode the bits of such a sequence. However, addressing information in at least one additional dimension is required to uniquely identify the spatial locations of those glyphs or any other glyphs that are included within a two dimensional glyph code pattern. These unique identifiers of the individual glyphs are referred to as "absolute addresses" because they identify the unique locations at which the individual glyphs reside within the glyph code pattern.

[0006] As is known, the spatial address (i.e., the absolute address) of almost any given glyph in an address space of the foregoing type is identifiable, at least to a first approximation, by a metric which specifies the offset distances (if any) between the nominal center of the given glyph and the non-parallel lines of glyphs that encode the respective maximal length bit sequences, where these offsets are measured parallel to the principal axes of the code pattern and are expressed in units of glyphs. In applications where the maximal length sequences are encoded by glyphs on orthogonal lines which align with the principal axes of a self-clocking glyph code pattern that is written on a regular rectangular lattice of centers, the above-described metric reduces to an x/y coordinate pair which identifies the spatial location of the given glyph with substantial precision in a standard Cartesian coordinate system. Unfortunately, however, the intersections of the lines on which these maximal bit length sequences are encoded tend to restrict the freedom of design choice if there are glyphs at any of these intersections.

[0007] It has been shown that this unwanted limitation on the explicit spatial addressing of glyphs in self-clocking glyph code patterns can be avoided if the glyphs are written on a lattice of centers of predetermined width and/or height, so that there are a known or determinable fixed number of glyphs on every line of the code pattern along at least one of those dimensions. More particularly the solution that has been proposed for this special class of code patterns is to encode a maximal length bit sequence (or a combination of interleaved, relatively prime maximal length bit sequences) in the glyphs in a raster-like pattern which is selected to cause the logically ordered bits of the sequence to spatially propagate from side-to-side along a known dimension (say, the width) of the code pattern which more gradually propagating along the other (e.g., the height) of the code pattern in, say, top-to-bottom order. This raster encoding of the maximal length bit sequence effectively "folds" the bit sequence into the code pattern module a known or computable number of glyphs (i.e., the number of glyphs that the code pattern contains along its known dimension). Accordingly, the spatial address of any given glyph within the code pattern can be computed in x/y coordinate space from the sequence index number of the bit that is

encoded by the glyph by dividing the sequence index number of the bit by the known glyph count/line modules.

[0008] Unfortunately, this raster encoding style solution to the spatial addressing problem is not only limited to a constrained class of code patterns, but also tends to be computationally costly. The computational costs of this approach are elevated because the subsequence bit length, N, over which the maximal length bit sequence must be unique scales as a function of the square root of the address space that is being served. This scaling is an inherent consequence of the fact that the number of unique phases (or "bit index positions") of a maximal length bit sequence is given by $N^2 - 1$.

[0009] Clearly, therefore, it would be desirable to have a more flexible technique for embedding logically ordered address information in some or all of the glyphs of self-clocking glyph code patterns for more computationally efficiently identifying the unique spatial locations of individual glyphs within such code patterns. Indeed, it would be desirable to parameterize such code patterns in N-dimensional space so that these address spaces, or fragments of them, can be used for hierarchically identifying two and three dimensional objects.

[0010] Human interpretable textual, graphical or mixed textual and graphical representations of the files that are accessible via this user interface are spatially registered in superimposed or juxtaposed relationship with respective addresses in this address space. These spatial addresses, in turn, are logically registered in a lookup table or the like with computer recognizable descriptions of the respective file names and, if needed, paths to the directories for the named files.

Fig. 1 illustrates a self-clocking glyph code pattern and a portion of its binary interpretation;

Fig. 2 is a block diagram that shows an example of a user interface implemented using glyph address carpets;

Fig. 3 is a block diagram of an image capture system that may be used to capture a user-selected portion of the glyph address carpet, and decode the glyphs in the captured portion;

Fig. 4 shows an embodiment of a user interface implementing hot zones;

Fig. 5 is a block diagram illustrating the selection process using camera pen 1710;

Fig. 6 is a block diagram illustrating a second type of selection process that may be used to implement a graphical user interface consistent with the principles of the invention

Fig. 7 illustrates an embodiment of the address codes encoded in the glyph address carpet 1732;

Fig. 8 and Fig. 9 form a flow chart showing the disambiguation and address decoding processes performed on the captured image area by computer 1712;

Fig. 10 illustrates a binary data matrix 2310 formed from a glyph lattice captured by camera pen 1710;

Fig. 11 is a flowchart showing the process for performing correlation steps 2116 and 2118;

Fig. 12 illustrates the analysis performed by computer 1712 to determine the address encoded by the glyphs of the captured image; and

Fig. 13 illustrates how the address carpet 1612 is divided into areas having respective ranges of addresses.

[0011] Turning now to the drawings, and at this point especially to Fig. 1, there is a more or less conventional self-clocking glyph code pattern 21 which is composed of elongated slash-like marks or "glyphs" 22 and 23 that are written on a generally regular rectangular lattice of centers on a suitable recording medium 24. Suitably, the glyphs 22 and 23 are printed by a printer (not shown) operating at 300 d.p.i. - 600 d.p.i. to write 4 pixel x 4 pixel - 7 pixel x 7 pixel representations of the glyphs 22 and 23 on regularly spaced centers that are distributed widthwise and lengthwise of the recording medium 24 to produce the rectangular code pattern 21. The glyphs of these fine grain glyph code patterns are not easily resolved by the unaided human eye when the code patterns are viewed under standard lighting conditions and at normal reading distances, so the code pattern 21 typically has a generally uniform gray scale appearance. Nevertheless, the glyph code is still capable of effectively communicating machine readable digital information. To carry out this function, the glyphs 22 and 23 usually are titled to the left and right, at about +45° and -45° with respect to the longitudinal dimension of the recording medium 24 to encode binary "1's" and "0's", respectively, as shown at 25.

[0012] In practice, the address space fragments 85-91 may be written directly on the objects 105-111, respectively, and/or on other substrates that are subsequently permanently or temporarily affixed to the objects. While a two dimensional address space can be employed for this application, more fully parameterized address spaces, such as the labeled address spaces $AL_1 \ldots AL_n$ have the advantage of having a hierarchical organization which may be employed, as shown, to hierarchically organize object identifiers 85-91 by object type or any other desired classification.

[0013] Fig. 2 is a block diagram that shows an example graphical user interface implemented using a glyph address carpet. The interface comprises two components: (1) sensory indicia and (2) a background which comprises a glyph address carpet 1612. The address carpet may be visible or invisible. The sensory indicia in Fig. 2 are preferably visual indicia, and in particular graphic elements, such as icon 1610. The glyph address carpet 1612 is used as the background wallpaper for the graphic elements. The glyph pattern that makes up the background is called an "address carpet" because the glyphs can be decoded to provide unique address information for each location.

[0014] The icons may, as shown in Fig. 2, be partially formed by glyphs that are part of the glyph address carpet. In the case of glyphtones (see, e.g., U.S. Patent No. 5,315, 098 to Tow) and serpentones (see, e.g., U.S. Patent No. 5,706,099 to Curry), the indicia may be substantially composed of the glyphs themselves. Because the glyphs encode address information, a portion of the glyphs can be read optically and decoded to determine a unique address for a local position.

[0015] The graphical user interface implemented using glyph address carpets can be implemented on any media capable of displaying the components of the interface. Thus, the interface could be implemented on hardcopy, such as paper, labels, physical objects, and photographic media; on dynamic displays such as a cathode ray tube (CRT) display or liquid crystal display (LCD); or on projected displays from fixed or dynamic media such as a slide projector, or television.

[0016] Although the embodiment described herein utilizes glyphs, the glyphs can alternatively be replaced by any system, visible or invisible, that provides a data address code. Whereas the embodiment described herein utilizes visual indicia, the visual indicia can alternatively be replaced by any sensory indicia that could serve to guide a user or machine to select a location while the machine readable data address codes provide an address logical reference. The embodiment described herein utilizes a paper substrate, but the paper substrate can alternatively be replaced by any media in which the data address codes can be embedded and read by a machine.

[0017] The sensory indicia may be embedded in the same media as the data address codes or may be embedded in other media. The sensory indicia can alternatively be projected onto glyphs, or glyphs can be projected onto sensory indicia.

[0018] Fig. 3 is a block diagram of an image capture system that may be used to capture a user-selected portion of a graphical user interface, such as glyph address carpet 12, and decode the glyphs in the captured portion. In one embodiment, computer system 1712 is a general purpose computer system, such as a conventional personal computer or laptop computer, that includes main memory 1716, read only memory (ROM) 1718, storage device 1720, processor 1722, and communication interface 1724, all interconnected by bus 1726. Bus 1726 also connects to display 1730, cursor control 1714, and frame capture 1728.

[0019] Image capture device 1710, which in this case is a camera pen, is connected to frame capture 1728 and mouse 1730. Camera pen 1710 transmits image information to frame capture 1728. In one embodiment, button 1714 of camera pen 1710 is wired to mouse 1730 so that when a user presses button 1714 a signal travels through the circuitry of mouse 1728 to cursor control 1714. The signal causes processor 1722 to run a program that directs frame capture 1728 to capture the image from camera pen 1710. In another embodiment, both the image line and signal line from camera pen 1710 are input directly into frame capture card 1728. The lines between camera pen 1710 and computer 1712 can be wired in any way that provides capture of the image from camera pen 1710.

[0020] The user makes a selection by placing camera pen 1710 on or near visual indicia on glyph address carpet 1732, and pressing button 1714. Pressing button 1714 causes camera pen 1710 to capture the portion of the address carpet under the tip of camera pen 1710, and transmit the image to computer 1712, via frame capture 1728, for analysis. The button or multiple buttons can be used for additional signaling, as in a double click, hold down.

[0021] Fig. 4 shows an embodiment of a user interface implementing hot zones. A hot zone is an area on or near the visual indica that defines selection locations that are equivalent to selecting the same visual indicia. In a preferred embodiment, the hot zone may encompass the visual indicia. For example, in Fig. 4, the icon David's DOC2 has a broken line area indicating a hot zone. If the user makes a selection within this hot zone, the icon "David's DOC2" will be selected. This allows the user to make a selection that is on or near the visual indicia. The selection is treated by the system as if the visual indicia was selected. The hot zone may be explicitly denoted by the visual indicia, including a visually distinct coloring in the address carpet, or may be implicit by appropriate proximity.

[0022] Fig. 4 also illustrates a preferred embodiment for the glyph address carpet coding scheme. Each glyph is either a forward slash or a backward slash. The orientation and spacing of the rows and columns of glyphs is indicated by O_{GX} and O_{GY}, respectively. As denoted by the As and Bs in the drawing, an A code runs on every other line, and is interlaced with a B code. Along diagonals running down and to the right, every A value is preferably the same. Similarly, along diagonals running down and to the left, every B value is preferably the same.

[0023] Fig. 5 is a block diagram illustrating the selection process using camera pen 1710. Each icon has an effective selection area indicated by a broken line hot zone 1910. The tip of camera pen 1710 covers an area indicated by 1914. The orientation of the tip of camera pen 1710 is denoted by the Y_C axis and X_C axis. To make a selection, the user places the tip of camera pen 1710 over the area to be selected. When the user presses button 1714, the image within area 1914 is captured. Computer 1712 analyzes the captured image to determine the location of the center 1912 of selection area 1914. After determining the location of center 1912, the location of center 1912 is used to look up a function corresponding to center 1912.

[0024] Fig. 6 is a block diagram illustrating a second type of selection process that may be used to implement a graphical user interface consistent with the principles of the invention. In this embodiment, camera pen 1710

has a pointing device attached to the tip, thus making the image capture area of the pen offset from the location pointed at by the user. For example, in Fig. 6, a user selects the icon "David's DOC" by pointing to the icon, but the camera pen 1710 image capture area 2014, with center 2012, is offset from the icon. In this case, computer 1712 must determine the actual selection based on 1) the image area 2014, and 2) the orientation of the selected area, and 3) the offset distance and direction from the image capture area from center 2012. The offset calculation utilizes the glyph lattice parameters from the captured image decoding process described below.

[0025] Returning to Fig. 3, in one embodiment, main memory 1716 is a random access memory (RAM) or a dynamic storage device that stores instructions executed by processor 1722. Main memory 1716 may also store information used in executing instructions. ROM 1718 is used for storing static information and instructions used by processor 1722. Storage device 1720, such as a magnetic or optical disk, also stores instructions and data used in the operation of computer system 1712.

[0026] Display 1730 may be a CRT or other type of display device. Cursor control 1714 controls cursor movement on display 1730. Cursor control 1714 may be, for example, a mouse, a trackball or cursor direction keys.

[0027] The system shown in Fig. 3 can be used to implement the glyph address carpet capture and translation system described herein. The apparatus and methods described herein may be implemented by computer system 1712 using hardware, software, or a combination of hardware and software. For example, the apparatus and methods described herein may be implemented as a program in any one or more of main memory 1716, ROM 1718, or storage device 1720. In one embodiment, processor 1722 executes programs which analyze captured portions of a glyph address carpet to determine address information encoded in the glyphs.

[0028] Such programs may be read into main memory 1716 from another computer-readable medium, such as storage device 1720. Execution of sequences of instructions contained in main memory 1716 causes processor 1722 to perform the process steps consistent with the present invention described herein. Execution of sequences of instructions contained in main memory 1716 also causes processor to implement apparatus elements that perform the process steps. Hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

[0029] The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 1722 for execution. Such a medium may take many forms, including but not limited to, non-volatile memory media, volatile memory media, and transmission media. Non-volatile memory media includes, for example, optical or magnetic disks, such as storage device 1720. Volatile memory media includes RAM, such as main memory 1716. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 1726. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infrared data communications.

[0030] Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic storage medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read and use.

[0031] Various forms of computer readable media may be involved in carrying one or more sequences of instructions to processor 1722 for execution. For example, the instructions may initially be carried on a magnetic disk or a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 1712 can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector coupled to appropriate circuitry can receive the data carried in the infrared signal and place the data on bus 1726. Bus 1726 carries the data to main memory 1716, from which processor 1722 retrieves and executes the instructions. The instructions received by main memory 1716 may optionally be stored on storage device 1720 either before or after execution by processor 1722.

[0032] Computer system 1712 also includes a communication interface 1724 coupled to bus 1726. Communication interface 1724 provides two way communications to other systems. For example, communication interface 1724 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. Communication may also be, for example, a local area network (LAN) card to provide communication to a LAN. Communication interface 1724 may also be a wireless card for implementing wireless communication between computer system 1712 and wireless systems. In any such implementation, communication interface 1724 sends and receives electrical, electromagnetic or optical signals that carry data streams representing various types of information.

[0033] The link between communication interface 1724 and external devices and systems typically provides data communication through one or more networks or other devices. For example, the link may provide a connection to a local network (not shown) to a host computer or to data equipment operated by an Internet Service Provider (ISP). An ISP provides data

communication services through the world wide packet data communications network now commonly referred to as the "Internet". Local networks and the Internet both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals between the networks and communication interface 1724, which carry the digital data to and from computer system 1712, are exemplary forms of carrier waves transporting the information.

[0034] Computer system 1712 can send messages and receive data, including program code, through the network(s) via the link between communication interface 1724 and the external systems and devices. In the Internet, for example, a server might transmit a requested code for an application program through the Internet, an ISP, a local network, and communication interface 1724.

[0035] Program code received over the network may be executed by processor 1722 as it is received, and/or stored in memory, such as in storage device 1720, for later execution. In this manner, computer system 1712 may obtain application code in the form of a carrier wave.

[0036] Fig. 7 illustrates an embodiment of the address codes encoded in the glyph address carpet 1732. More particularly, Fig. 7 illustrates one section 2110 of the glyph address carpet. The addresses are encoded by alternating rows of "A" address code sequences and "B" address code sequences. The position along each sequence in each row should be unambiguously determinable from a predetermined length subsequence. For example, an N-bit shift register maximal length code can be uniquely determined in position from an N-bit subsequence. Each address code sequence is a fifteen bit sequence, with the A sequence indexed running left to right, and the B sequence indexed running in a reverse direction, right to left. Each row of A code sequences is offset by two glyph positions relative to the previous and next row of A addresses. Similarly, each row of B code sequences is offset in the opposite direction by two positions. Thus, the encoding scheme has two key characteristics: parallel rows including two sets of one-dimensional unique address codes and relative offsets among members of the two sets so that the offset between each pair from the two sets is unique. This establishes two-dimensional unique address locations.

[0037] Computer 1712 decodes address information encoded in the glyphs by analyzing the captured image area in two steps. Ideally, the user places camera pen 1710 over a portion of glyph address carpet 1732 and capture an image angularly aligned as shown in the pattern of bits shown in Fig. 7. In reality, however, the user variably orients the camera pen 1710 over an area of interest, so the pattern could be oriented anywhere from 0° to 360°. Therefore, the computer 1712 must first determine the orientation of the image as part of decoding and interpreting the address information.

[0038] The orientation of the image is determined by analyzing the captured image. This process is called disambiguation. (See, e.g., U.S. Patent No. 5,521,372 to Hecht, et al.). After determining the proper orientation of the image, computer 1712 decodes the address of the selected location in the address carpet. The disambiguation and address decoding processes performed by computer 1712 will now be described in greater detail.

[0039] Fig. 8 and Fig. 9 form a flow chart showing the disambiguation and address decoding processes performed by computer 1712 on the captured image area. Computer 1712 begins the disambiguation process by image processing the captured portion of the address carpet to find a glyph seed. A glyph seed is a first-identified glyph having readable glyphs around it. Once a glyph seed has been found, the glyph lattice can be determined by processing neighbors of the glyph seed. (See Appendices A and B). The glyphs are then decoded as 1's or 0's, which are filled into a binary data matrix having rows and columns corresponding to the glyph lattice rows. The orientation may still be ambiguous with respect to 90° and 180° rotations.

[0040] Fig. 10 illustrates a binary data matrix (BDM) 2310 formed from a glyph lattice captured by camera pen 1710. The BDM has locations corresponding to the glyph lattice. Thus the size of the BDM corresponds closely to the size of the glyph lattice.

[0041] Each location of the glyph lattice is analyzed to determine which value should be placed in the corresponding location of the BDM. Initially, the BDM is filled with a value, for example ϕ, which indicates that no attempt has been made to read the glyph. Once the glyph corresponding to a particular location has been analyzed, the ϕ is replaced by a value indicating the result of the glyph analysis.

[0042] In Fig. 10, a B indicates a border location; an X indicates that no interpretable glyph was found at the corresponding location of the glyph lattice; an E indicates a glyph at the edge of the captured image portion; a 0 indicates a back slash glyph; and a 1 indicates a forward slash glyph. The area of the matrix corresponding the captured image is filled with 0's and 1's, the edge is bounded by Es, and the Xs correspond to locations that have no readable glyphs. In practice, however, although the BDM will generally have a similar pattern, the values will often not be as evenly distributed. For example, a glyph location within the captured image area might result in an X if the glyph has been obliterated. Several values have been drawn with circles and squares around them to illustrate the two separate code sequences that are staggered in opposite directions.

[0043] When a user makes a selection, the user might orient the camera pen in virtually any direction on the user interface. The captured image could be oriented at any angle. Thus, even though step 2210 allows computer 1712 to derive a BDM of 0's and 1's from the captured image, it is uncertain whether the BDM is oriented at 0°

(i.e., correctly oriented), 90°, 180°, or 270° relative to the original code pattern in the glyph address carpet from which the image was captured. Until the orientation of the captured image is determined, it is not possible to derive the correct address code from the BDM. The orientation could be provided by auxiliary information such as physical system constraints. However, the orientation can be uniquely determined directly from the address codes.

[0044] After converting the glyphs to 0's and 1's, a reference glyph location is selected (step 2211). This location may be chosen in a variety of ways, but is typically a location which represents the selection. For example, the reference glyph location could be at the center of the BDM.

[0045] After the image has been converted to a BDM, it is processed by computer 1712 (step 2212). The original BDM developed from the captured image is referred to as BDM1. Computer 1712 makes a copy of BDM1 and rotates the copy clockwise 90° to form a second binary data matrix, BDM2 (step 2214). By rotating BDM1 by 90°, the rows of BDM1 'become the columns of BDM2, and the columns of BDM1 become the rows of BDM2. Additionally, all bit values in BDM2 are flipped from 0 to 1, and 1 to 0, because a 45° slash glyph rotated 90° appears as the opposite state of the non-rotated glyph.

[0046] Computer 1712 then performs a correlation separately on the odd and even rows of BDM1 (step 2216) to determine whether code in the rows are staggered forward or backward. The correlation is also performed for the odd and even rows of BDM2 (step 2218). The correlation is performed over all the rows of each BDM, and results in correlation value C1 for BDM1 and correlation value C2 for BDM2.

[0047] Fig. 11 is a flowchart showing an embodiment of correlation steps 2216 and 2218 of Fig. 9. The process determines a correlation value for every other line of a BDM along diagonals in each direction, and sums the row correlation values to form a final correlation value for the odd or even rows. The process is performed on the odd rows of BDM1 to form correlation value $C1_{ODD}$ for BDM1, the even rows of BDM1 to form correlation value $C1_{EVEN}$ for BDM1, the odd rows of BDM2 to form correlation value $C2_{ODD}$ for BDM2, the even rows of BDM2 to form correlation value $C2_{EVEN}$ for BDM2. The BDM that is oriented at 0° or 180° will have a larger $C_{ODD} + C_{EVEN}$ than the other BDM.

[0048] Computer 1712 first inputs the BDM (step 2410), and then copies every other row to a temporary matrix (step 2412). Identical processes are then performed for diagonals to the right and to the left. Steps 2414, 2416, 2418, 2420, 2422 and 2424 process the diagonals to the right. For example, in Fig. 12 the steps correlate along the diagonals moving from the upper left to lower right. First, row count N and correlation value C_RIGHT are each initialized to zero (step 2414). Row N is shifted two places to the right, and correlated with the next row (step 2416). C_N is then set to this value (step 2418). C_RIGHT is then set to C_RIGHT + C_N (step 2420), and N is incremented (step 2422). If row count N >= Nmax, where N is the number of odd or even rows in the BDM, then the process proceeds to step 2426. If N is not greater than or equal to Nmax, the process continues at step 2416. Thus, after the process has correlated each adjacent row, the correlation value C_RIGHT indicates the strength of the correlation along the diagonals to the right.

[0049] The steps on the right of Fig. 11 are similar to steps 2414, 2416, 2418, 2420, 2422 and 2424, but process diagonals running from the upper right to lower left to develop correlation value C_LEFT. After correlating the right and left diagonals to determine C_RIGHT and C_LEFT, a final correlation value C is determined by subtracting C_LEFT from C_RIGHT. For example, if odd rows for BDM1 are processed, the C value becomes $C1_{ODD}$ for BDM1.

[0050] The process steps of Fig. 11 are performed for the odd and even rows of BDM1 and the odd and even rows of BDM2. From this information, the correlation value C1 for BDM1 is set to $C1_{EVEN} + C1_{ODD}$ (as determined by Fig. 11 for the rows of BDM1), and the correlation value C2 for BDM2 is set to $C2_{EVEN} + C2_{ODD}$ (as determined by Fig. 11 for the rows of BDM1).

[0051] Fig. 12 illustrates why the correlations determine which way the codes in every other row are shifting. For example, as indicated by the circled A1s along the diagonals running to the right, the codes along the diagonal starting at A1 in the first position of the second row should have the same value in every other row along the diagonal, except for erasures and errors. Similarly, as indicated by the boxed B1s, the codes along the diagonal starting at B1 in the upper right corner should have the same value in every other row along the diagonal, except for erasures or errors. This is true for each value along the diagonal in the odd rows running respectively from B2, B3, ... in the top row. Thus, the strong correlations along the diagonals running down and to the left on the odd rows, and the strong correlations along the diagonals running down and to the right on the even rows indicate that the codes in the even rows are shifting to the right, and the codes in the odd rows are shifting to the left.

[0052] For each BDM, therefore, four correlation values are developed: 1) odd rows, right to left, 2) odd rows, left to right, 3) even rows, right to left and 4) even rows, left to right. From these correlation values, the strongest correlation value for the even rows, and strongest correlation value for the odd rows is chosen, and these become C_{EVEN} and C_{ODD} for that BDM (steps 2216 and 2218). C_{EVEN} and C_{ODD} are then added to form a final C correlation value for that BDM. As discussed above with respect to step 2220, the BDM with the strongest correlation value is the BDM that is oriented at either 0° or 180° because of the relative orientation of the codes in the odd and even rows. Thus, two aspects of the cho-

sen BDM are now established: which direction every other line of codes is staggered, and that the BDM is oriented horizontally, at either 0° or 180°. Another correlation process, step 2230 is performed to determine which direction the code in each line runs (as opposed to which way the code is staggered).

[0053] The codes in the odd lines are staggered in one direction, and the codes in the even lines are staggered in the other. This staggering property of the code, in conjunction with knowing the respective codes that run in the odd lines and even lines, allows determination of the proper 0° orientation of the BDM.

[0054] Returning to Fig. 8, if C1 is greater than C2 (step 2220), then BDM1 is selected for further processing. C1 being greater than C2 indicates that the one-dimensional codes of BDM1 are most strongly correlated and are, therefore, oriented at either 0° or 180° (step 2222). If C2 is greater than C1, then BDM2 is selected for further processing, because the higher correlation indicates that BDM2 is oriented at either 0° or 180° (step 2224). Thus, the correct BDM has been found. Before determining the address location of the captured image, however, computer 1712 must first determine whether the selected BDM is at 0° (i.e., oriented correctly), or rotated by 180°.

[0055] Fig. 9 is a flowchart showing the steps of how computer 1712 determines the address of the captured area of the glyph carpet. Preferably, bit positions along a diagonal in the BDM, when the BDM is oriented at 0°, have the same value at every other row. The image capture process and interference from the visual indicia, however, may result in errors and erasures in the BDM data. To reduce the impact of these errors and erasures, computer 1712 performs a majority vote along each diagonal of the odd rows in the direction the odd rows are staggered, and repeats the majority vote process for even rows along the each diagonal in the direction the even rows are staggered (step 2225). This results in a first code sequence for the odd rows and a second code sequence for the even rows. To the extent the majority vote correctly determines each bit position, the first and second code sequences should match a subsequence of the original pseudo noise address sequence respectively corresponding to the odd or even set of rows.

[0056] Computer 1712 then retrieves the original pseudo noise address code (Get Code 1) for rows staggered forward (step 2226), and retrieves the original pseudo noise address code for (Get Code 2) for rows staggered backward (step 2228). Using the original pseudo noise address code for each code set A and B, and the code from the majority voting, computer 1712 performs four cross correlations (step 2230) to establishes the best match of the glyph sequence with the PN sequence location for the odd and even rows.

[0057] More particularly, two adjacent rows of the BDM closest to the reference element chosen in step 2211 are correlated with the respective complete PN sequences that created the original address carpet. The PN sequences could be identical. A forward and backward correlation is performed for each row. The four correlations develop four pairs of peak correlation and position values:

1) P1, V1, respectively representing the peak correlation value and corresponding position for Code 1 correlated backwards with the complete PN sequence;

2) Q1, U1, respectively representing the peak correlation value and corresponding position for Code 2 correlated forwards with the complete PN sequence;

3) P2, V2, respectively representing the peak correlation value and corresponding position for Code 1 correlated forwards with the complete PN sequence; and

4) Q2, U2, respectively representing the peak correlation value and corresponding position for Code 2 correlated backwards with the complete PN sequence.

[0058] The U_i and V_i, position values, where i = 1 or 2, corresponding to the peak magnitudes are used to determine the X and Y values corresponding to the reference element chosen in step 2211. Thus, if (P1 + Q1) > (P2 + Q2) (step 2232), then UI and V1 are used to calculate the X, Y position of the reference glyph location chosen in step 2211 (step 2236). If (P1 + Q1) <= (P2 + Q2) (step 2232), then U2 and V2 are used to calculate the X, Y position of the reference glyph location chosen in step 2211 (step 2234). The address information is determined in accordance with the following equations:

$$X = (V_i - U_i + \text{length of full code})/2$$

$$Y = (V_i + U_i - \text{length of full code})/2$$

[0059] The calculated X, Y position is then returned (step 2238). Note that diagonals correspond to constant values of U and V, respectively, while rows and columns correspond to constant X and Y. Note also that U and V could be used directly as address parameters.

[0060] Thus, an X, Y value associated with the reference point chosen in step 2211 has been determined. Using this information, computer 1712 associates the X, Y coordinates with a logical reference, or a combination of a logical reference and a control signal (e.g., a button click), with a particular operation to be performed. For example, the X, Y coordinates could be used as an index into a table of operations performable by computer 1712 or other device under direction of computer 1712. The X, Y coordinates could be associated with a file open command that opens a file associated with an icon located near the X, Y coordinates in the address space.

Virtually any operation that can be performed by computer 1712 could be associated with a particular X, Y coordinate, or range of X, Y coordinates.

[0061] Fig. 13 illustrates an embodiment of a user interface in which address carpet 12 is divided into hot zones having respective ranges of addresses corresponding to a graphic object which in turn corresponds to a system object or function. Computer 1712 uses the X,Y address determined from the image capture to determine which area of address carpet 12 was selected by the user. For example, if the address is X=1052 and Y=32, which is in the upper left area of address carpet 12, then the user has selected CloseWin.avi. Computer 1712 utilizes a table that associates X, Y coordinates to one or more functions that are to be performed based on the X, Y coordinates. For example, in the above example, X=1052 and Y=32 would be associated with the one or more functions to be performed when a user selects CloseWin.avi. Activation could be signaled to computer 1712 by button 1714, such as a click or multiple click. Alternatively, activation could be signaled to computer 1712 by holding the image capture steady for a certain time. The precision of location selection is one glyph unit and the resolution of differentiated selection can be as fine as one glyph unit. Resolution can be extended to a fraction of a glyph unit provided the capture selection device can be referenced to the address carpet to within a fraction of a glyph interval. For example, a camera pixel is typically fraction of a glyph interval.

[0062] Methods, systems and articles of manufacture consistent with the present invention therefore facilitate implementation of a graphical user interface using glyph address carpets. The foregoing description of an implementation of the invention has been presented for purposes of illustration and description. It is not exhaustive and does not limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practicing of the invention.

Claims

1. Apparatus for implementing a user interface, comprising:

   sensory indicia embodied in a media (1732);
   a two-dimensional embedded data address code (2110);
   a reader for reading a portion of the two-dimensional address code (1710); and
   a translator (1722) for translating the portion into a discrete pointer associated with the sensory indicia.

2. A method for implementing a user interface using sensory indicia embedded in a media and a two-dimensional embedded data address code, comprising:

   reading a portion of the two-dimensional address code (2210); and
   translating the portion into a discrete pointer (2238).

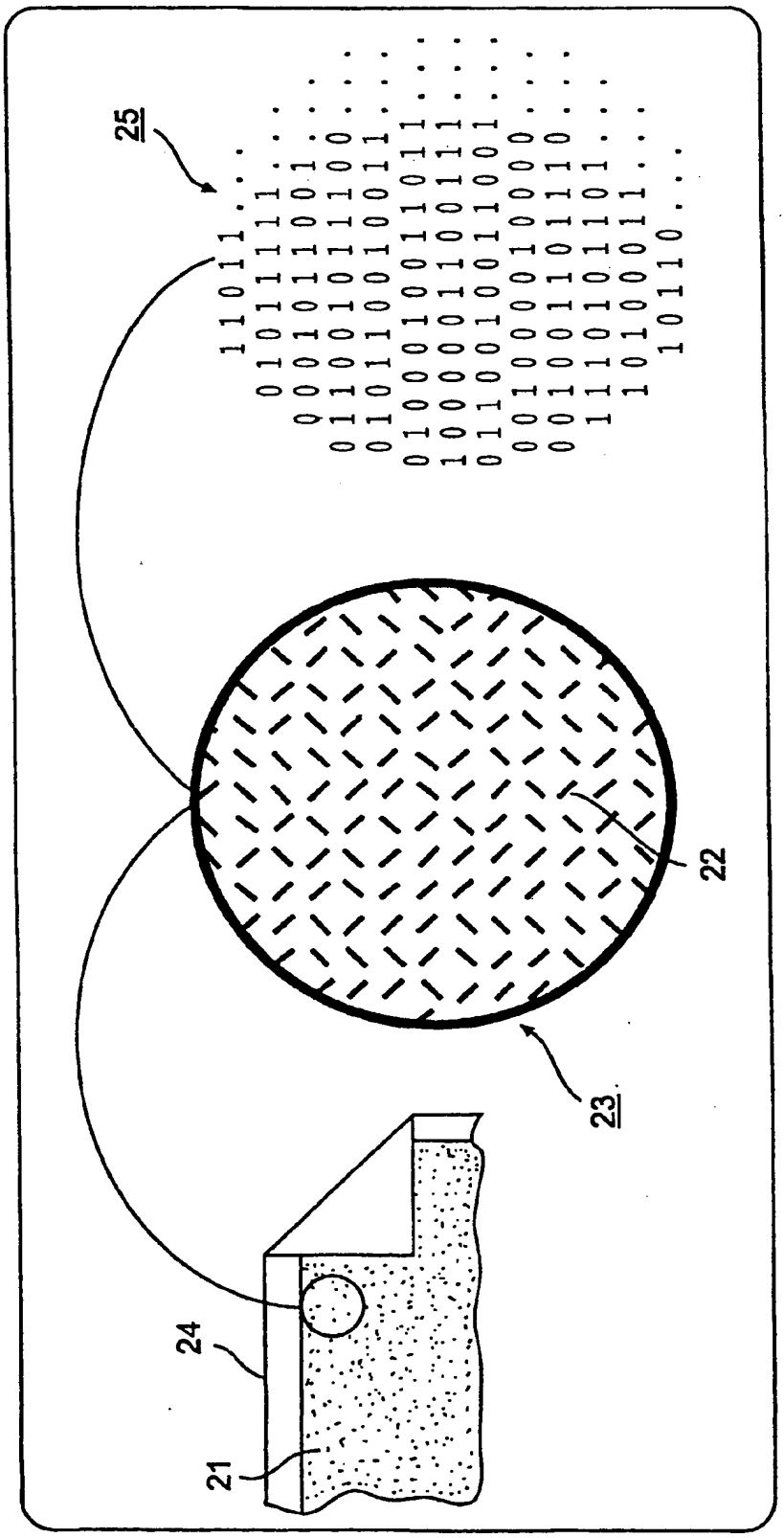

FIG. 1

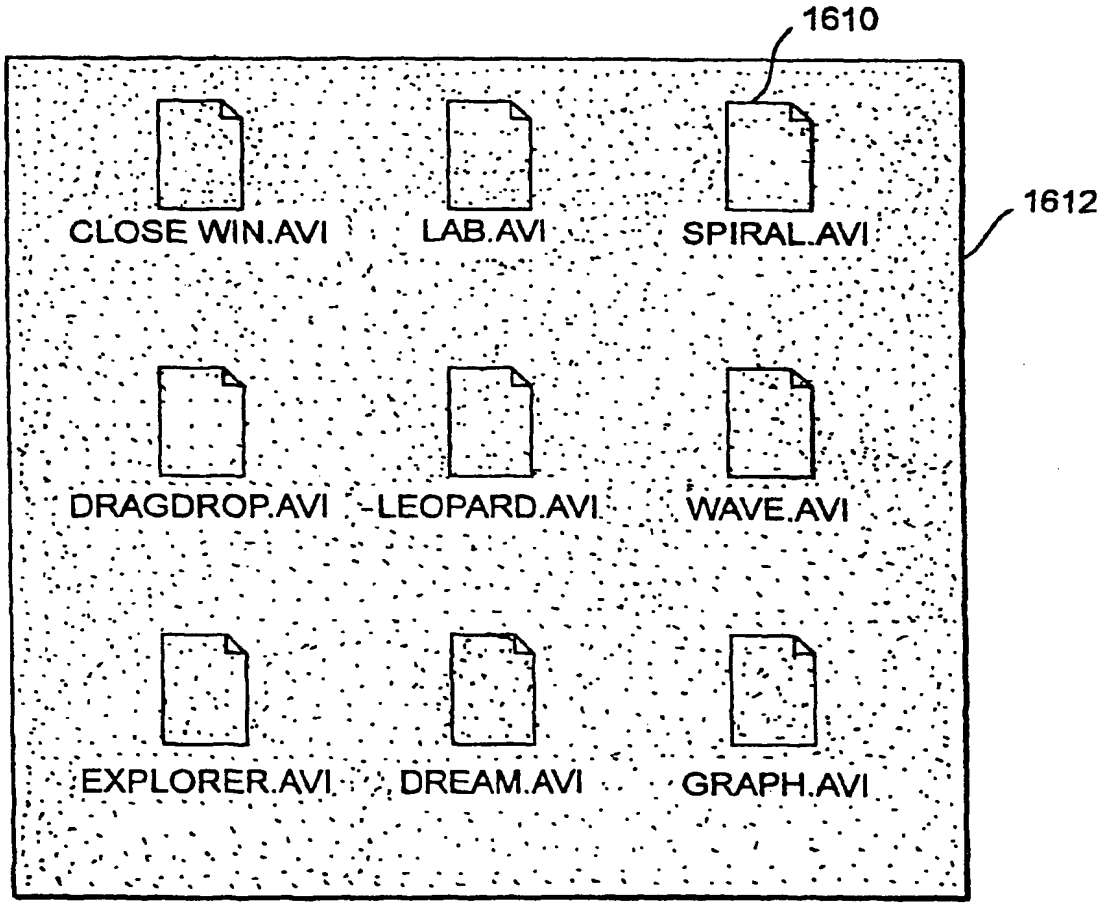
1610
1612
CLOSE WIN.AVI
LAB.AVI
SPIRAL.AVI
DRAGDROP.AVI
LEOPARD.AVI
WAVE.AVI
EXPLORER.AVI
DREAM.AVI
GRAPH.AVI

FIG. 2

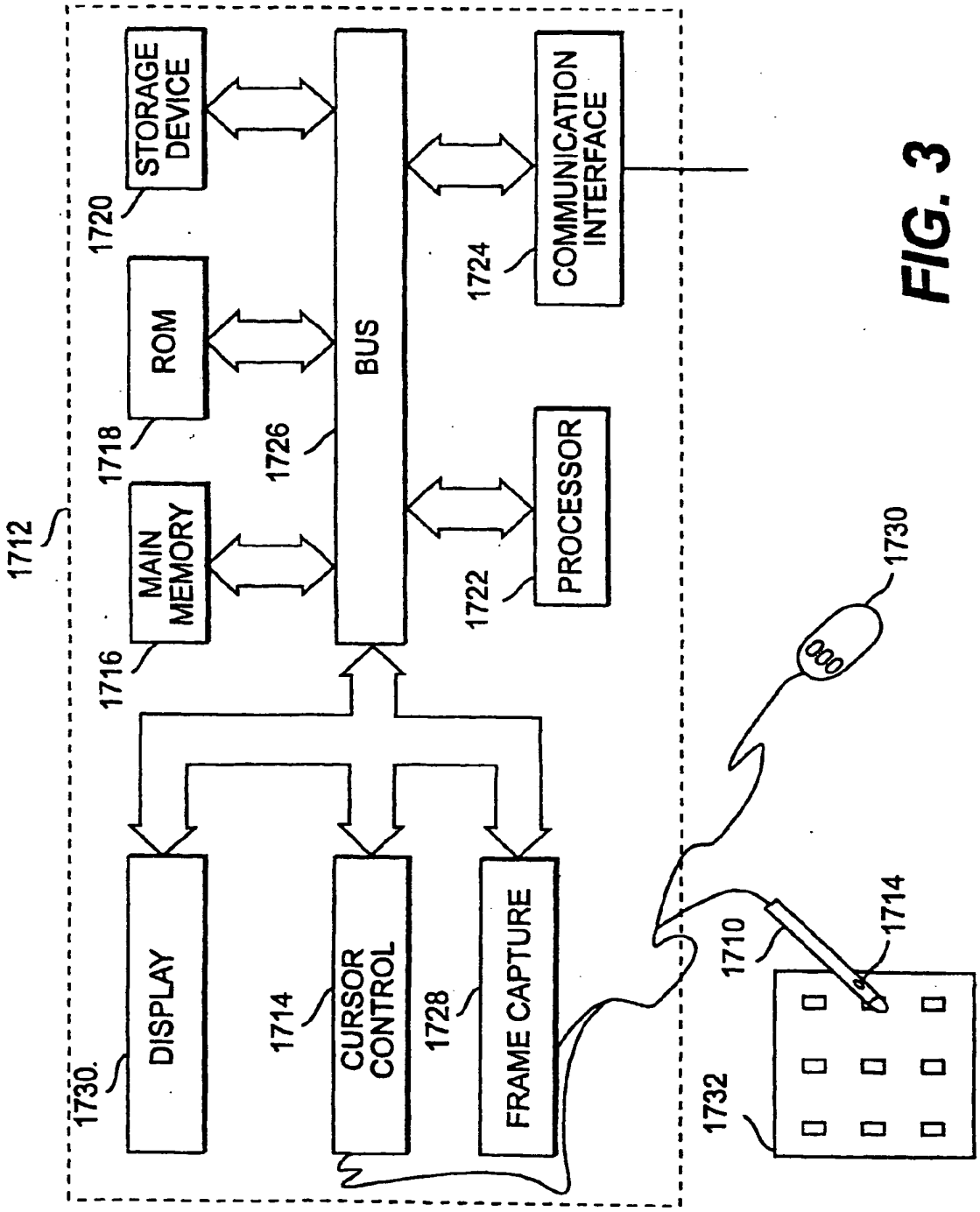
1712
STORAGE DEVICE
1720
ROM
1718
MAIN MEMORY
1716
BUS
1726
COMMUNICATION INTERFACE
1724
PROCESSOR
1722
1730
DISPLAY
1730
CURSOR CONTROL
1714
FRAME CAPTURE
1728
1710
1714
1732

FIG. 3

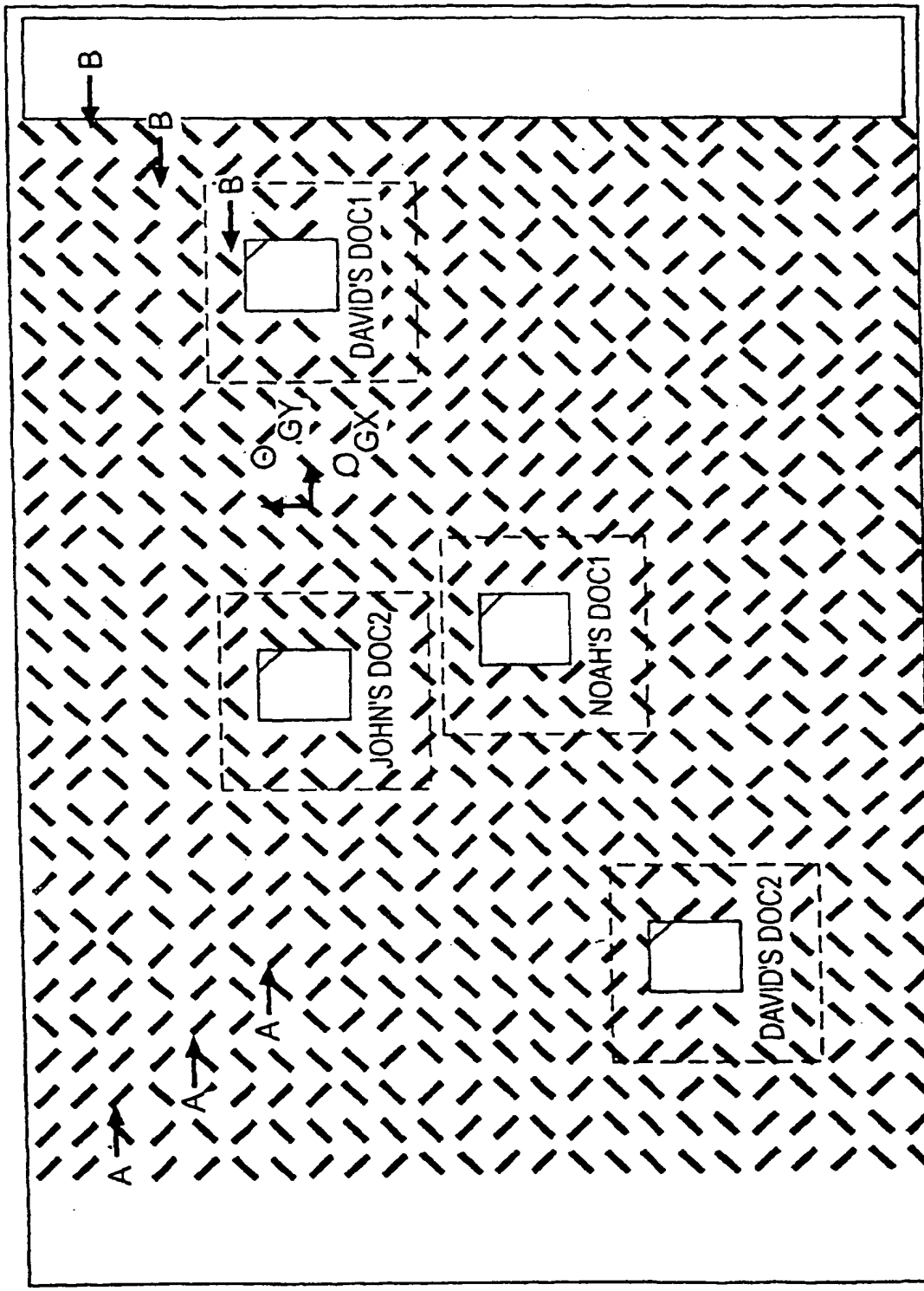

FIG. 4

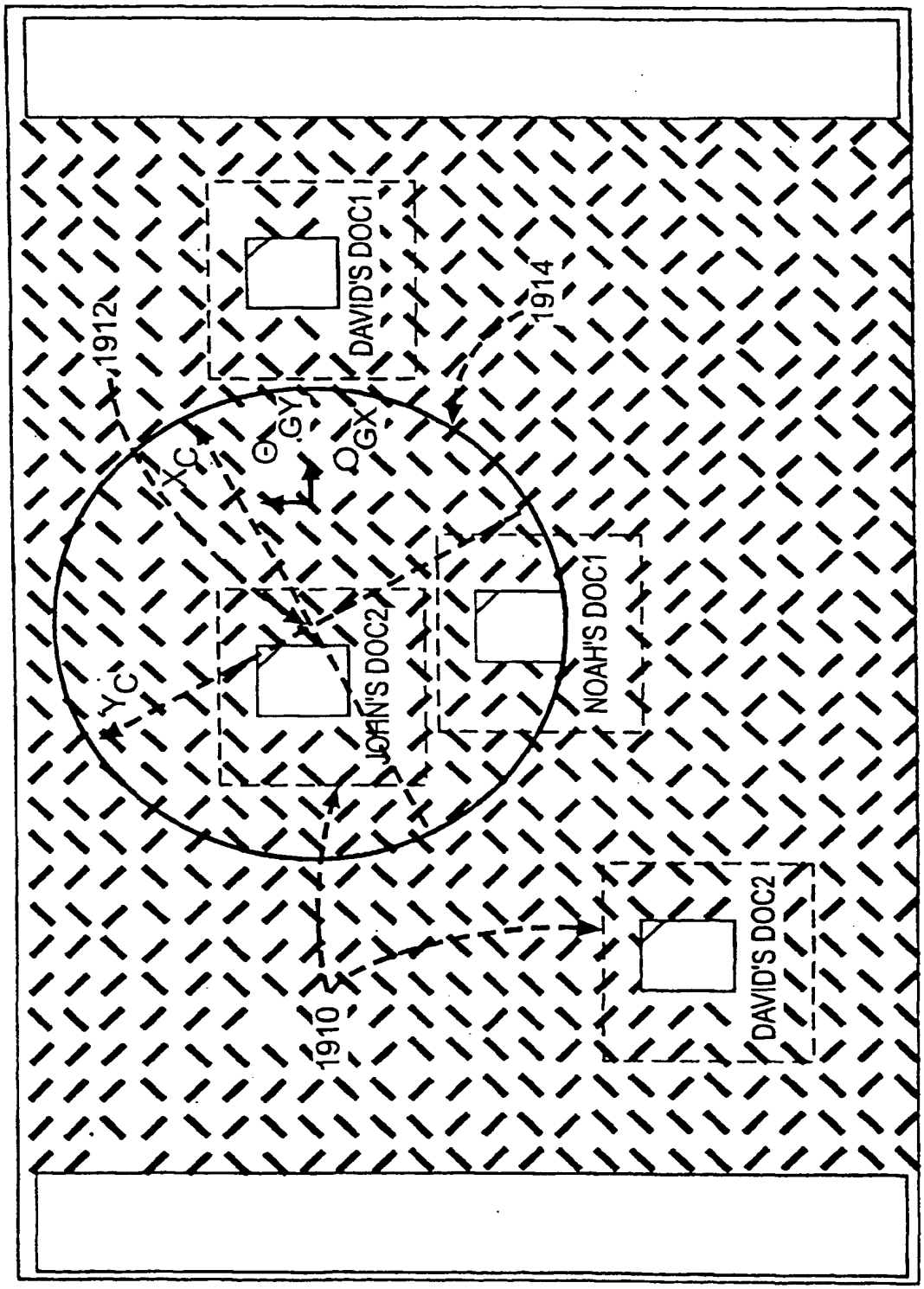
1912
DAVID'S DOC1
1914
Y_C
X_C
Θ_{GY}
Q_{GX}
JOHN'S DOC2
NOAH'S DOC1
1910
DAVID'S DOC2

FIG. 5

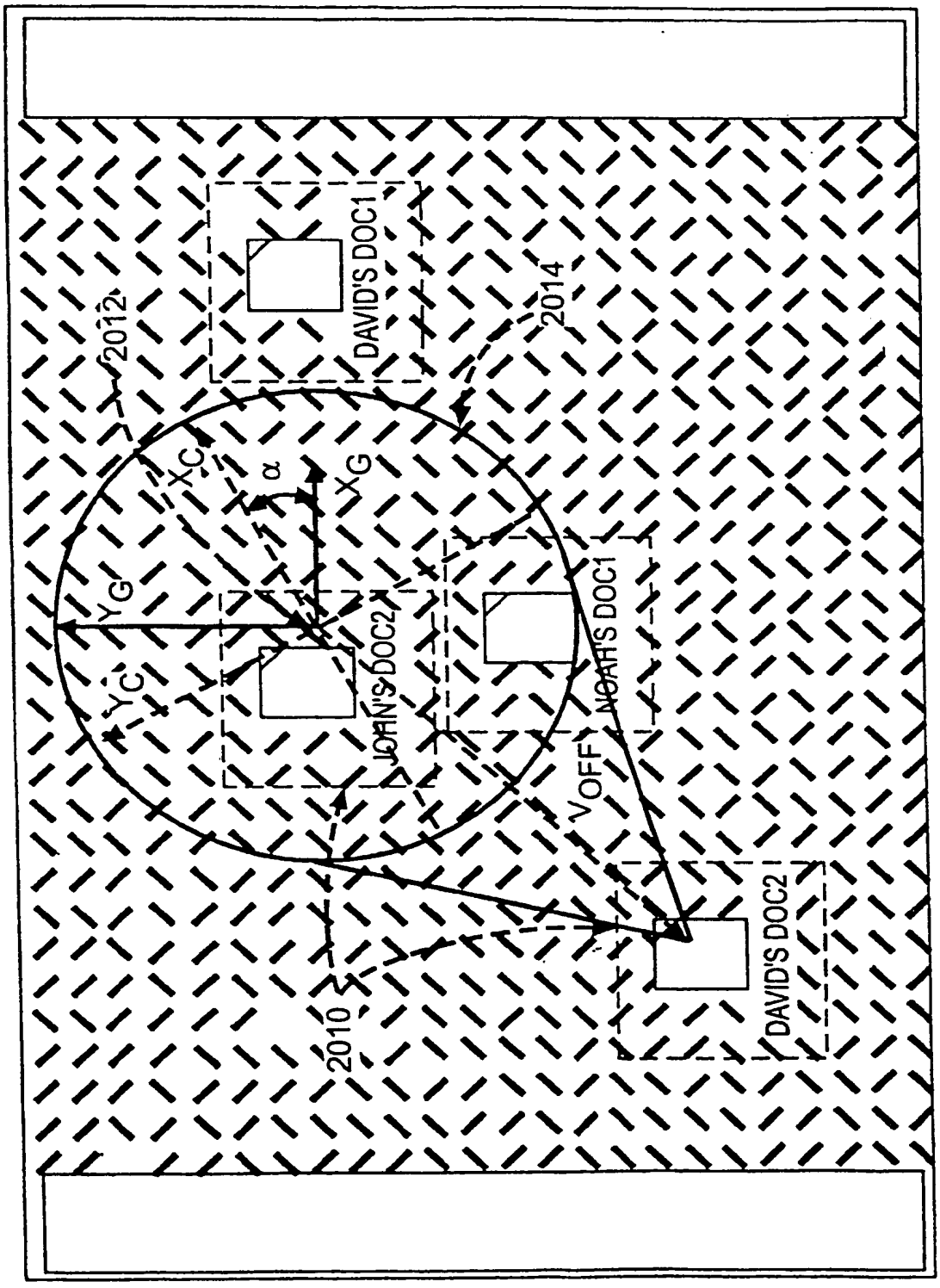
DAVID'S DOC1
2012
2014
X_C
α
X_G
Y_G
Y_C
JOHN'S DOC2
NOAH'S DOC1
V_{OFF}
2010
DAVID'S DOC2

FIG. 6

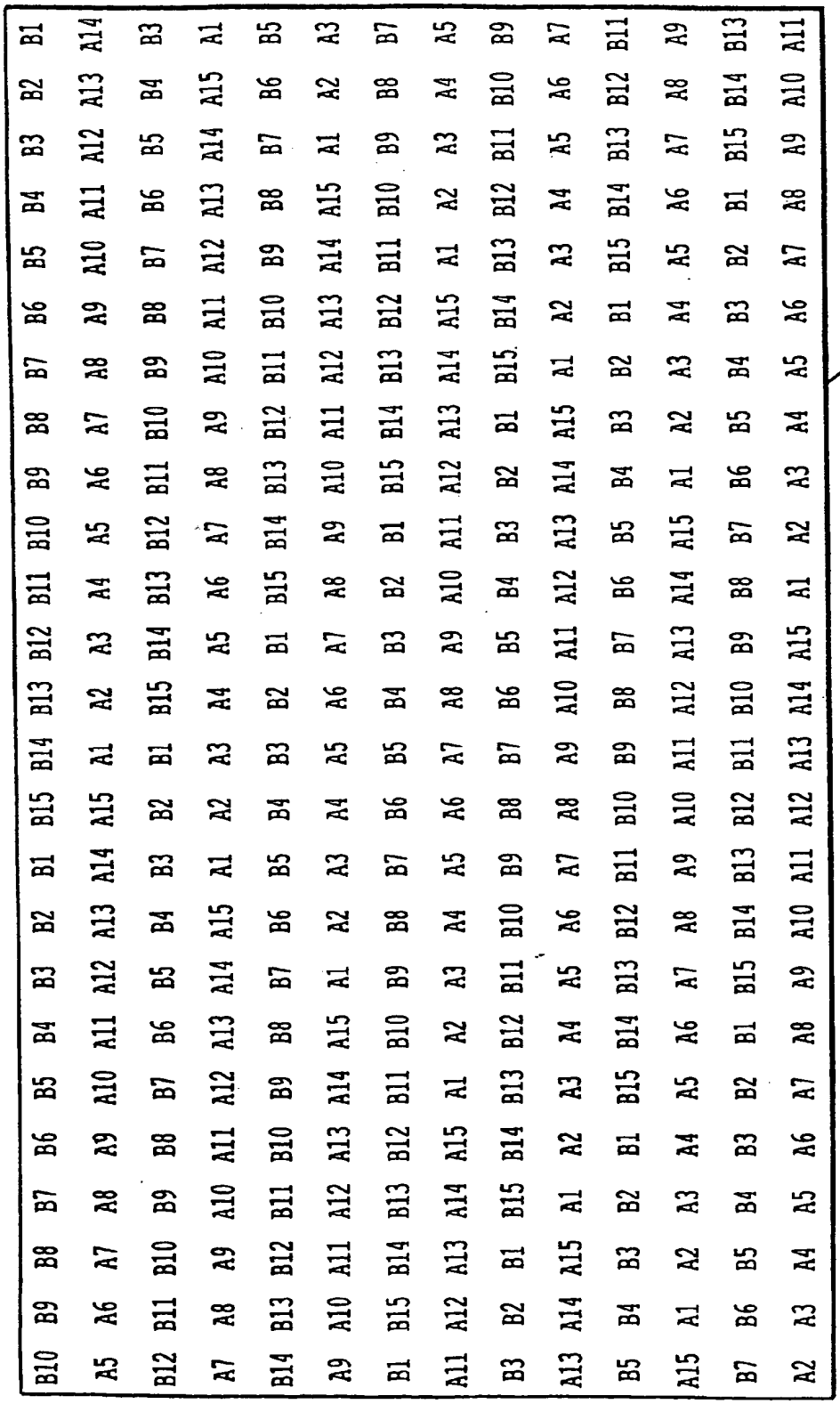
B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1
A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1
B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3
A7 A8 A9 A10 A11 A12 A13 A14 A15 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1
B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5
A9 A10 A11 A12 A13 A14 A15 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1 A2 A3
B1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B15 B14 B13 B12 B11 B10 B9 B8 B7
A11 A12 A13 A14 A15 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1 A2 A3 A4 A5
B3 B2 B1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B15 B14 B13 B12 B11 B10 B9
A13 A14 A15 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1 A2 A3 A4 A5 A6 A7
B5 B4 B3 B2 B1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B15 B14 B13 B12 B11
A15 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1 A2 A3 A4 A5 A6 A7 A8 A9
B7 B6 B5 B4 B3 B2 B1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B15 B14 B13
A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
2110

FIG. 7

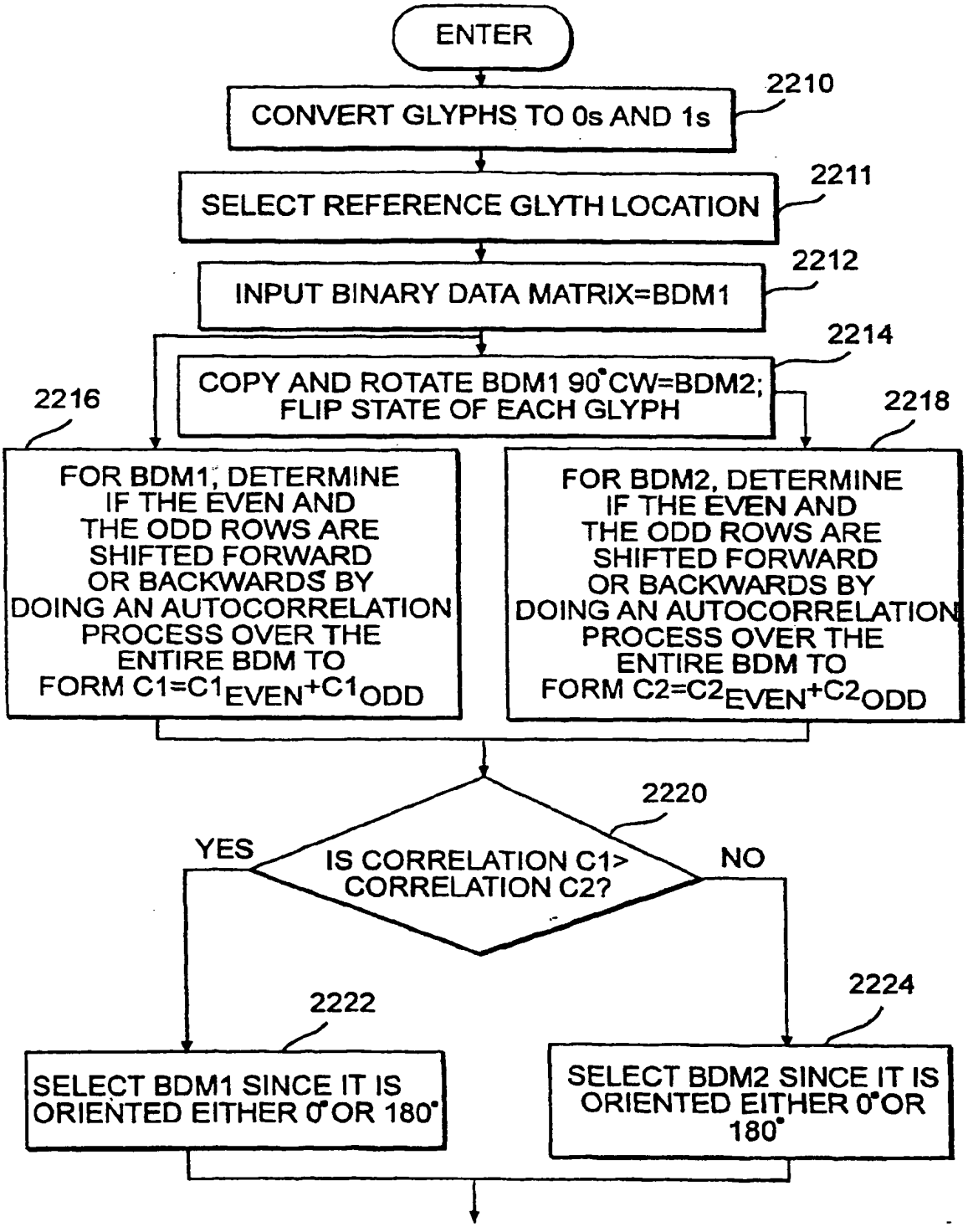
ENTER
2210
CONVERT GLYPHS TO 0s AND 1s
2211
SELECT REFERENCE GLYTH LOCATION
2212
INPUT BINARY DATA MATRIX=BDM1
2214
COPY AND ROTATE BDM1 90°CW=BDM2; FLIP STATE OF EACH GLYPH
2216
FOR BDM1, DETERMINE IF THE EVEN AND THE ODD ROWS ARE SHIFTED FORWARD OR BACKWARDS BY DOING AN AUTOCORRELATION PROCESS OVER THE ENTIRE BDM TO FORM C1=C1EVEN+C1ODD
2218
FOR BDM2, DETERMINE IF THE EVEN AND THE ODD ROWS ARE SHIFTED FORWARD OR BACKWARDS BY DOING AN AUTOCORRELATION PROCESS OVER THE ENTIRE BDM TO FORM C2=C2EVEN+C2ODD
2220
IS CORRELATION C1> CORRELATION C2?
YES
NO
2222
SELECT BDM1 SINCE IT IS ORIENTED EITHER 0°OR 180°
2224
SELECT BDM2 SINCE IT IS ORIENTED EITHER 0°OR 180°
TO FIG. 9

FIG. 8

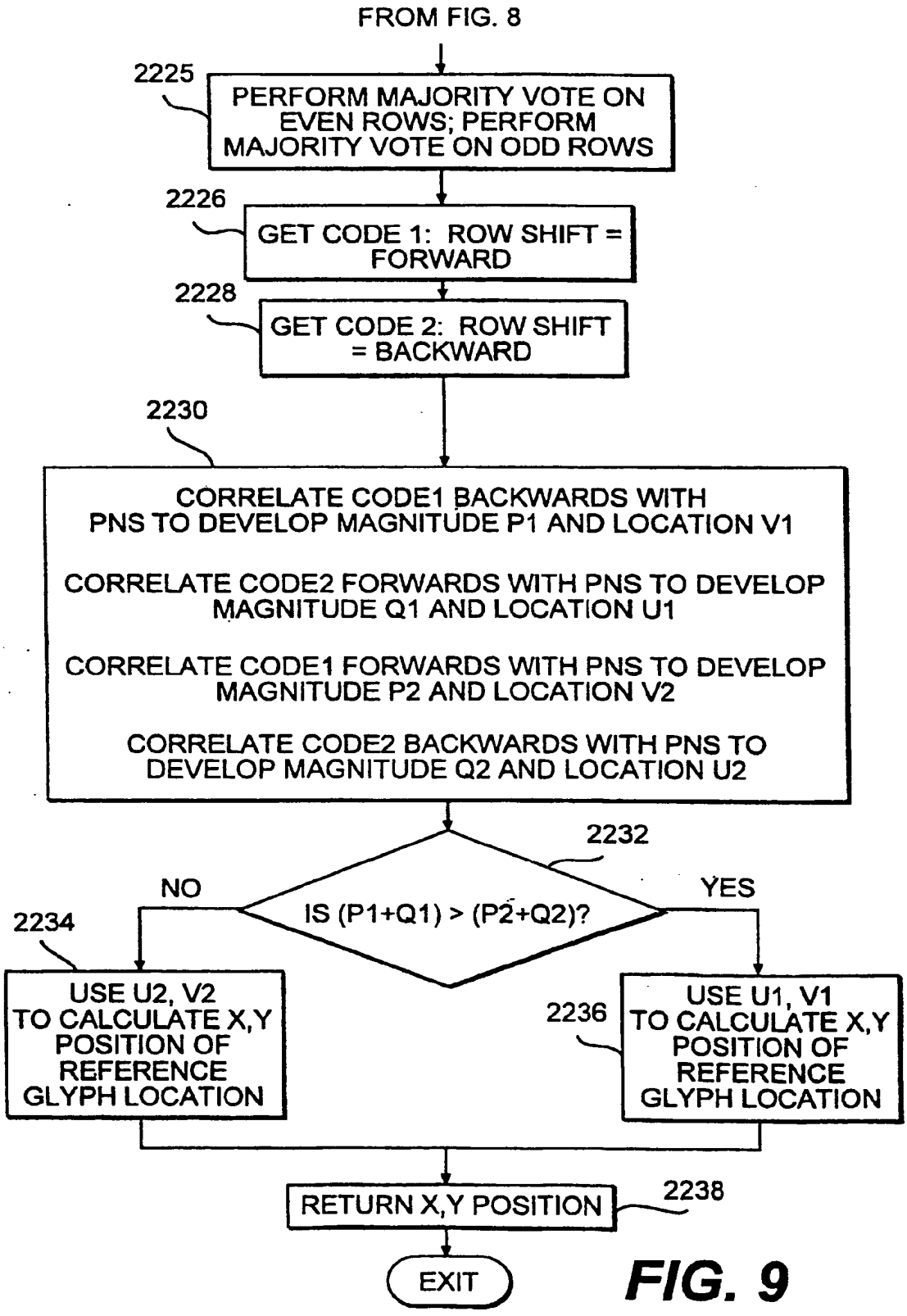
FROM FIG. 8
2225
PERFORM MAJORITY VOTE ON EVEN ROWS; PERFORM MAJORITY VOTE ON ODD ROWS
2226
GET CODE 1: ROW SHIFT = FORWARD
2228
GET CODE 2: ROW SHIFT = BACKWARD
2230
CORRELATE CODE1 BACKWARDS WITH PNS TO DEVELOP MAGNITUDE P1 AND LOCATION V1
CORRELATE CODE2 FORWARDS WITH PNS TO DEVELOP MAGNITUDE Q1 AND LOCATION U1
CORRELATE CODE1 FORWARDS WITH PNS TO DEVELOP MAGNITUDE P2 AND LOCATION V2
CORRELATE CODE2 BACKWARDS WITH PNS TO DEVELOP MAGNITUDE Q2 AND LOCATION U2
2232
IS (P1+Q1) > (P2+Q2)?
NO
YES
2234
USE U2, V2 TO CALCULATE X,Y POSITION OF REFERENCE GLYPH LOCATION
2236
USE U1, V1 TO CALCULATE X,Y POSITION OF REFERENCE GLYPH LOCATION
RETURN X,Y POSITION
2238
EXIT

FIG. 9

B	B	B	B	B	B	B	B	B	B	B	B	B	B
B	X	X	X	E	E	0	0	E	E	X	X	X	B
B	X	X	E	1	0	1	1	0	0	1	E	X	B
B	X	E	1	0	0	1	1	1	1	0	E	X	B
B	E	0	1	(0)	0	1	0	1	1	0	0	E	B
B	E	0	0	1	1	1	1	0	[1]	1	0	0	B
B	0	0	0	0	1	(0)	0	1	0	1	1	0	B
B	0	1	1	1	1	0	[1]	1	0	0	0	1	B
B	E	1	0	0	0	0	1	(0)	0	1	0	1	B
B	E	1	1	0	[1]	1	0	0	0	1	1	E	B
B	X	1	0	1	0	0	0	0	1	(0)	E	X	B
B	X	E	[1]	1	0	0	0	1	1	1	E	X	B
B	X	X	X	E	0	1	0	0	E	E	X	X	B
B	B	B	B	B	B	B	B	B	B	B	B	B	B

FIG. 10

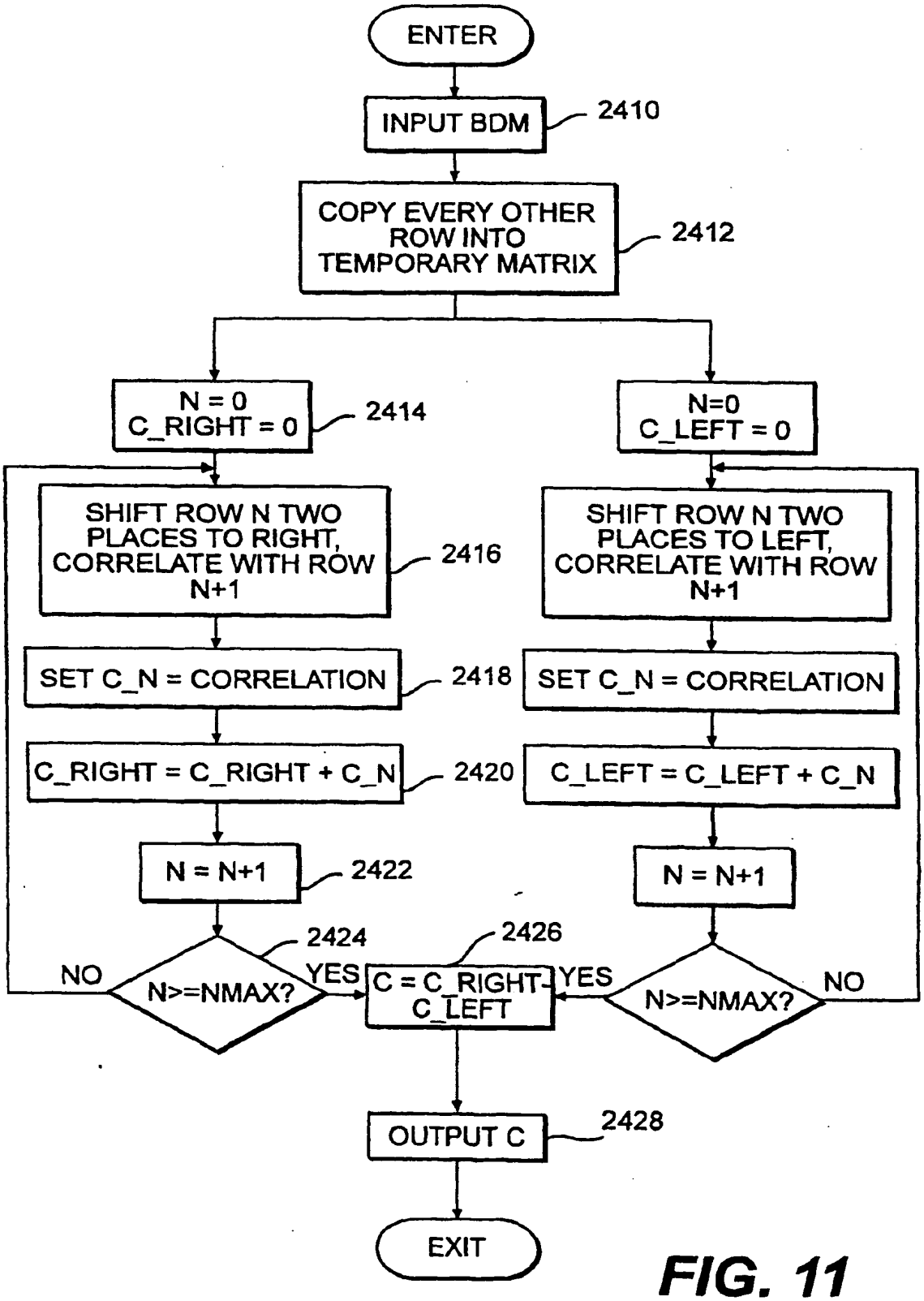
ENTER
INPUT BDM
2410
COPY EVERY OTHER ROW INTO TEMPORARY MATRIX
2412
N = 0
C_RIGHT = 0
2414
N=0
C_LEFT = 0
SHIFT ROW N TWO PLACES TO RIGHT, CORRELATE WITH ROW N+1
2416
SHIFT ROW N TWO PLACES TO LEFT, CORRELATE WITH ROW N+1
SET C_N = CORRELATION
2418
SET C_N = CORRELATION
C_RIGHT = C_RIGHT + C_N
2420
C_LEFT = C_LEFT + C_N
N = N+1
2422
N = N+1
2424
2426
NO
N>=NMAX?
YES
C = C_RIGHT C_LEFT
YES
N>=NMAX?
NO
OUTPUT C
2428
EXIT

FIG. 11

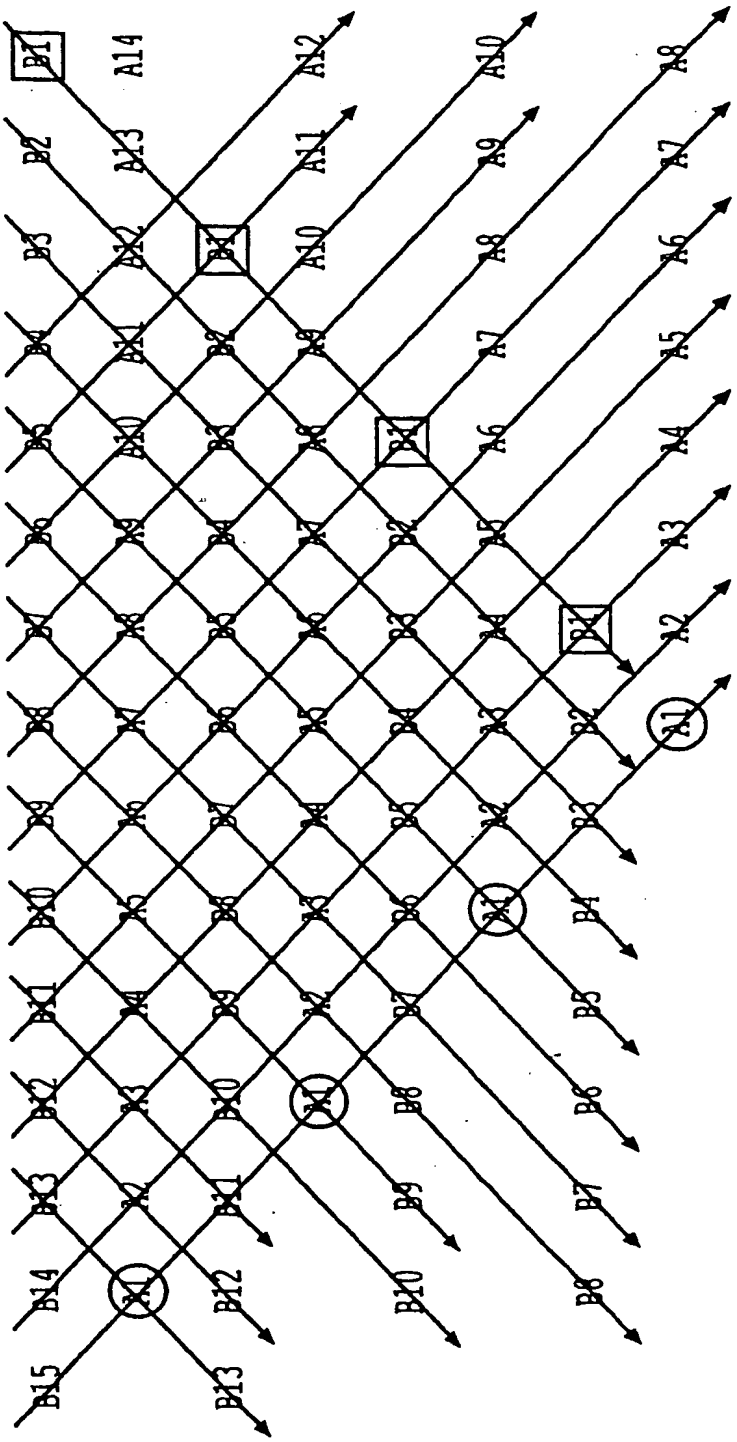

FIG. 12

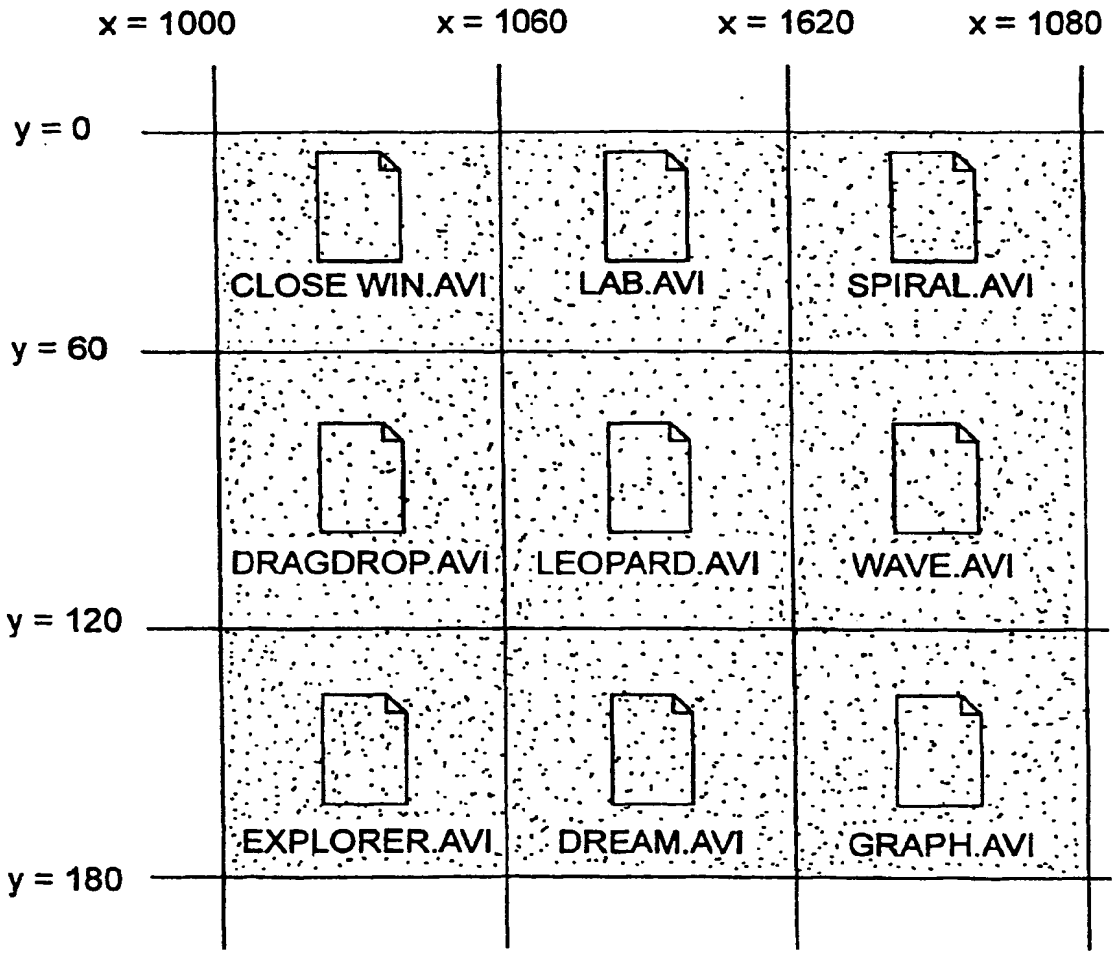
x = 1000
x = 1060
x = 1620
x = 1080
y = 0
y = 60
y = 120
y = 180
CLOSE WIN.AVI
LAB.AVI
SPIRAL.AVI
DRAGDROP.AVI
LEOPARD.AVI
WAVE.AVI
EXPLORER.AVI
DREAM.AVI
GRAPH.AVI

FIG. 13